BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号
特開2000－122916
（P2000－122916A）
(43)公開日　平成12年４月28日(2000.4.28)

(51)Int.Cl.⁷	識別記号	ＦＩ	ﾃｰﾏｺｰﾄﾞ(参考)
Ｇ０６Ｆ　12/00	５７２	Ｇ０６Ｆ　12/00	５７２Ａ

審査請求　未請求　請求項の数１　ＯＬ　（全 10 頁）

(21)出願番号　特願平11－286660

(22)出願日　平成11年10月７日(1999.10.7)

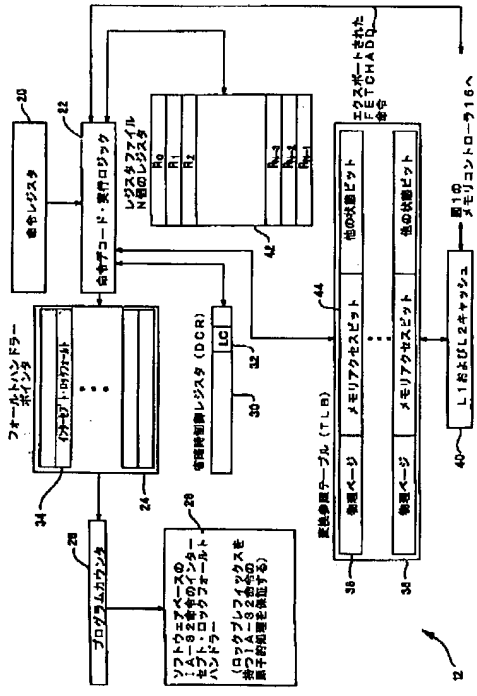
(31)優先権主張番号　１７０１３７
(32)優先日　平成10年10月12日(1998.10.12)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　599142110
エマージング・アーキテクチャーズ・エルエルシー
Ｅｍｅｒｇｉｎｇ　Ａｒｃｈｉｔｅｃｔｕｒｅｓ，Ｌ．Ｌ．Ｃ．
アメリカ合衆国95014カリフォルニア州クパーチノ、プルーンリッジ・アベニュー　19447　ヒューレット・パッカード・カンパニー内
(74)代理人　100081721
弁理士　岡田　次生

最終頁に続く

(54)【発明の名称】　原子的更新処理を実行する方法

(57)【要約】
【課題】ソフトウェアが、ハードウェアにより提供される最高のパフォーマンスの原子的更新方法にアクセスできるようにする。
【解決手段】エクスポート可能な６４ビットのFETCHADD命令を定義する。それぞれの仮想メモリページは、ライトバック方式を使用するキャッシュ可(WB)、キャッシュ不可(UC)、キュッシュ不可でエクスポート可(UCE)のいずれかのメモリ属性を持つ。FETCHADD命令が実行され、WBに設定された属性のページにあるメモリ位置がアクセスされると、ＣＰＵはそのメモリ位置を含むキャッシュラインの排他的使用を得ることによりFECHADDを原子的に実行する。UCEに設定された属性のページのメモリ位置がアクセスされると、メモリコントローラのような中央ロケーションにFETCHADD命令をエクスポートすることにより、ＣＰＵはFETCHADDを原子的に実行する。



1

【特許請求の範囲】

【請求項１】メモリ属性フィールドをアクセスして、原子的更新処理によりアクセスされるメモリ位置が、エクスポート可能な命令をサポートしているかどうかを判断するステップと、

前記原子的更新処理によりアクセスされるメモリ位置がエクスポート可能な命令をサポートしているならば、該原子的更新処理を中央ロケーションにエクスポートするステップと、

前記原子的更新処理によりアクセスされるメモリ位置がエクスポート可能な命令をサポートしていないならば、キャッシュ・コヒーレンシー機構を使用して原子的更新処理を実行するステップと、

を含む原子的更新処理を実行する方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、コンピュータシステムにおけるメモリアクセス操作に関する。より具体的には、本発明は、典型的にはセマフォにアクセスするのに使用される原子的メモリ更新処理に関する。

【０００２】

【従来の技術】コンピュータシステムにおいては、２つ以上のプロセスが同じリソースに対して競合することがよくある。例えば、２つ以上のプロセスが、特定のコマンドシーケンスをビデオコントローラに書き込もうとすることがある。これらのプロセスは、１つの中央処理装置（ＣＰＵ）により実行されることもあれば、マルチプロセッサコンピュータシステムの２つ以上のＣＰＵにより実行されることもある。ここでは、「ＣＰＵ」及び「プロセッサ」という用語を、互いに取り替え可能に使用する。

【０００３】複数のプロセスが、同時に１つのリソースをアクセスすることができないので、コンピュータのオペレーティングシステムは、リソースへのアクセスをスケジュールするなんらかの機構を提供しなければならない。当該技術分野で知られている一般的な機構の１つとして、「番号取得（take-a-number）」スケジューリング・アルゴリズムがある。このアルゴリズムは、１人の店員の手があくのを待っている顧客の集団に多少似ている。顧客は店に入るときに番号を受け取る。店員がその番号を呼ぶと、その顧客は店員のサービスを受けることができる。

【０００４】これに類似したものとして、「番号」をプロセスに提供する機構は、当該技術分野ではセマフォとして知られている。典型的には、セマフォはメモリ位置に記憶される。セマフォをアクセスしようとするプロセスは、最初にメモリ位置を読み出し、このメモリ位置から読み出した値をインクリメントし、結果をそのメモリ位置に記憶し戻す。メモリ位置から読み出された値は、そのプロセスの「番号」の役割を果たし、メモリ位置に

2

記憶し戻された結果は、そのリソースをアクセスしようとする次のプロセスの次の「番号」の役割を果たす。特定の「番号」の保持者がリソースにアクセスしてもよいことをオペレーティングシステムが示すとき、その「番号」を持つプロセスがアクセスを行う。

【０００５】「番号取得」スケジューリングアルゴリズムが正確に作動するには、メモリ読み出し、インクリメントおよびメモリ書き込み処理が、「原子的」に発生しなければならない。言い換えると、第１のプロセスがメモリ位置を読み出した時点から、第１のプロセスがインクリメントした値をメモリ位置に記憶し戻す時点までの間は、セマフォを保持するメモリ位置を第２のプロセスが読み出す機会があってはならないということである。もし第２のプロセスによるそのような読み出し処理が発生すると、第１および第２のプロセスはそれぞれ同じ「番号」を持つことになり、リソースへのアクセスを同時に試みようとすることがある。

【０００６】セマフォ操作が原子的に発生するのを確実にすることは、バスに連結される他の装置が直接記憶アクセス（ＤＭＡ）処理を行わない単一ＣＰＵのコンピュータシステムにおいては比較的簡単なことである。例えば、３２ビットのIntel（商標）のアーキテクチャ（ＩＡ－３２）は、Intel i486™ 、Pentium（商標）、Pentium Pro、Pentium IIおよびCeleron™ のＣＰＵにより使用され、「ＸＡＤＤ（exchange and add；交換および加算）」命令を含んでいる。この命令を使ってセマフォを含むメモリ位置をアクセスするとき、ＸＡＤＤ命令は通常以下のように用いられる。

ＸＡＤＤ　宛先メモリ位置，ソースレジスタ

【０００７】この命令は、宛先メモリ位置およびソースレジスタに含まれる値の合計を一時レジスタに記憶し、宛先メモリ位置の内容をソースレジスタに記憶し、一時レジスタの内容を宛先メモリ位置に記憶する。従って、命令が実行されるときに値「１」がソースレジスタに記憶されていると、命令が完了した時に宛先メモリ位置の値は「１」だけインクリメントし、宛先メモリ位置にもともとあった値はソースレジスタに記憶される。命令が完了するまでは割込みが処理されることは無く、またこの例のコンピュータシステムが単一ＣＰＵ（他の装置は、ＤＭＡ処理を行わない）であるので、ＸＡＤＤ命令によって実行される「読み出し－変更－書き込み（read-modify-write；リードモディファイライト）」処理の間は、他のプロセスはセマフォにアクセスすることができない。したがって、セマフォ処理は原子的に発生する。ＩＡ－３２のＸＣＨＧ（exchange；交換）命令及びＣＭＰＸＣＨＧ（compareおよびexchange；比較および交換）命令もまた、セマフォへの原子的アクセスを確実にするのに広く用いられている。

【０００８】マルチプロセッサコンピュータシステムおよびＤＭＡ処理を実行するデバイスを備えるシステムに

おいては、第1のCPUがインクリメントしてセマフォをメモリ位置にと書き戻す前に、第2のCPUまたはデバイスがセマフォにアクセスしようとすることがあるので、原子性を保証するのがより複雑になる。このようなコンピュータシステムでは、バスのロック機構またはキャッシュのコヒーレンシー機構のいずれかを使用することにより原子性が提供される。これらの機構を詳細を述べる前に、CPUのキャッシュメモリの処理を最初に考えるのが有用である。

【0009】キャッシュメモリは、メインメモリの内容のサブセットを保持する比較的小容量で高速のメモリである。例えば、Pentium（商標）IIのCPUをベースとしたコンピュータシステムは、レベル1（L1）のキャッシュをCPUと同じ集積回路（IC）上に有しており、レベル2（L2）のキャッシュをCPUと同じモジュールではあるが異なるIC上に有している。L1キャッシュはL2キャッシュより小さく、より高速である。メインメモリの内容は、キャッシュラインと呼ばれる単位でキャッシュメモリに記憶される。Pentium IIのCPUでは、L1およびL2キャッシュのキャッシュラインの大きさが32バイトである。

【0010】Intel（商標）i486™のCPUは、「ライトスルー（write-through）」のL1キャッシュを採用する。このようなキャッシュにおいては、CPUからのメモリ書き込みが、キャッシュおよびメインメモリに同時に書込まれる。Intel PentiumのCPU以降、Intelのプロセッサは、「ライトバック（write-back）」のキャッシュをサポートしている。ライトバックキャッシュにおいては、CPUからのメモリ書き込みがキャッシュにのみ書込まれる。その後、キャッシュ機構が、そのメモリ書き込みが実際にメインメモリにコミットされたかどうか（および、いつコミットされたか）を判断する。これにより、メインメモリがビジーでなくなるまでメインメモリへの書き込みを遅らせることができるので、性能（パフォーマン）が上がる。さらに、メモリオペランドをメインメモリに書き戻す前に、メモリオペランドが何回か変わることがある。また、メモリにキャッシュラインを書き戻す前に、キャッシュラインの変更を完全に組み立てる機会がキャッシュに与えられるが、これは当該技術分野ではコウレシング（coalescing；併合）として知られている。

【0011】キャッシュ・コヒーレンシー機構は、CPUキャッシュおよびメインメモリに記憶されたメモリ内容が確実にコヒーレンス（一貫性）に保たれるようにする。例えば、第1のCPUのキャッシュが、メインメモリにまだ書き戻されていない、変更された（即ち「ダーティな(dirty)」）内容を持つキャッシュラインを含んでおり、第2のCPUが、メインメモリから対応するメモリ位置を読み出そうと試みる場合、キャッシュ・コヒーレンシー機構は、メインメモリに現在記憶された正しくない内容ではなく、第1のCPUのキャッシュからの正しい内容が、確実に第2のCPUに提供されるようにする。キャッシュ・コヒーレンシー機構は、これを幾つかの方法で実現することができる。1つの手法は、単純に第1のCPUのキャッシュに対し、変更されたキャッシュラインをメインメモリに強制的に書き戻させることである。他の手法は、第2のCPUのキャッシュが、第1のCPUのキャッシュに対する変更を「スヌープ（snoop；監視する）」できるようにすることにより、第1のCPUのキャッシュで行われた変更で、第2のCPUのキャッシュを継続的に更新できるようにする。

【0012】さらに、CPUは、キャッシュラインが「共用（shared）」または「専有（exclusive）」としてロードされるよう要求することができる。共用キャッシュラインはCPUにより変更することができず、従ってキャッシュラインの内容が変更されないことがわかっているような状況（例えば、プログラムコード）で有利に使用される。専有（または、代わりに「専用（private）」とも言う）キャッシュラインは、CPUにより変更することができる。典型的には、「ダーティビット（dirty-bit）」が、専有キャッシュラインに関連しており、内容が変更されたかどうかを示す。ダーティビットが設定され、キャッシュラインが変更されたことを示すならば、キャッシュラインをメインメモリに書き戻さなくてはならない。ダーティビットがクリアされ、キャッシュラインが変更されていないことを示すならば、メインメモリに書き戻されたものとしてキャッシュラインを廃棄することができる。通常、いずれの時点においても、1のみのCPUが特定のキャッシュラインを専有として保持することができる。

【0013】原子性の話に戻ると、初期のIA－32のCPUは、キャッシュ不可のメモリまたはライトスルー方法を使ってキャッシュされたメモリにセマフォを記憶することにより、そしてセマフォにアクセスするときに「バスロック（bus lock）」を発行することにより、原子性を提供する。バスロックは、セマフォ処理によって必要とされる「読み出し－変更－書き込み」トランザクションの間、1つのCPUが確実にバスの排他的所有権を持つようにする。この方法では、他のCUPがセマフォを含むメモリ領域にアクセスする必要がないとしても、「読み出し－変更－書き込み」トランザクションが完了するまでの間は、すべての他のCPUが、バスにアクセスすることからブロックされるので、パフォーマンスにかなり重い負担をかける。様々な相互接続構造を使用するハイエンドのマルチプロセッサシステムにおいては、「バス」という概念が完全に消えてしまうことがあり、したがって「バスロック」という概念も完全に消えてしまうことがあるということに注意されたい。例えば、4つのプロセッサから成るポッド（pod）を持ち、1つのポッドにおけるそれぞれのプロセッサが従来のバ

スを介して結合され、ポッドのそれぞれがリング・トポロジーで相互接続されているマルチプロセッサシステムにおいては、１つのポッドにおけるＣＰＵが、他のポッドにおけるバスをロックすることが通常できない。

【００１４】後のＩＡ－３２ＣＰＵは、キャッシュ・コヒーレンシー機構を介して原子性を提供する。ＣＰＵがセマフォをアクセスするとき、ＣＰＵのＬ１キャッシュが、セマフォを保持するメモリ位置を含むキャッシュラインの専有使用を要求する。従って、トランザクション中に他のＣＰＵがセマフォにアクセスできる可能性無しに、ＣＰＵは、セマフォ処理により必要とされる「読み出し－変更－書き込み」トランザクションを実行することができる。従って、他のＣＰＵは引き続きバスにアクセスすることができるので、引き続きメモリにアクセスすることができる。他のＣＰＵに対してアクセス可能でないメインメモリ領域だけが、セマフォ処理を実行するＣＰＵのキャッシュに専有として保持されるキャッシュラインであるので、本質的に、「キャッシュ内（in-cache）」の原子的更新が、「アドレスロック（address lock）」を介して実行される。そのキャッシュライン全体が専有として保持されるので、１つのキャッシュラインに複数のセマフォを記憶しない方が望ましい場合が多いということに注意されたい。

【００１５】このキャッシュ・コヒーレンシーを介した原子性の提供は、バスロックを介してキャッシュ・コヒーレンスを提供するよりもかなり良いパフォーマンスを提供するが、「セマフォのキャッシュラインのスラッシング」によってパフォーマンスがなお制限されることがある。セマフォのキャッシュラインのスラッシングは、２つ以上のＣＰＵが同じリソース、よって同じセマフォについて継続的に競合する時に発生する。したがって、それぞれのＣＰＵがセマフォを含むキャッシュラインの排他制御を得ようと継続的に試み、そのキャッシュラインが継続的にそれぞれのＣＰＵのキャッシュにロードされて書き込まれる。通常、ＣＰＵがセマフォを含むキャッシュラインに対する専有アクセス権を得るために待っている間は、そのＣＰＵの処理は進行することができない。

【００１６】従来技術において、大型マルチプロセッサシステムの中には、ＦＥＴＣＨＡＤＤ（fetch and add（フェッチおよび加算））命令を用いてこの問題に対処してきたものがある。「ＦＥＴＣＨＡＤＤ」命令に関連する「インクリメント」処理は、メモリコントローラのような中央ロケーションにエクスポートされる。したがって、ＣＰＵが、メモリ位置に記憶されたセマフォを参照するＦＥＴＣＨＡＤＤ命令を実行するとき、メモリコントローラは、メモリ位置に記憶されたセマフォ値をそのＣＰＵに提供する。さらに、メモリコントローラはセマフォをインクリメントし、その結果をそのメモリ位置に記憶し戻す。従って、ＣＰＵが、セマフォを含むメモリ位置に書き込む必要が無いので、ＣＰＵは、セマフォを含むキャッシュラインへの専有アクセスを獲得する必要はなく、それによりセマフォのキャッシュラインのスラッシングが取り除かれる。加えて、複数のセマフォが、パフォーマンスを犠牲にすることなくキャッシュラインの境界内に存在することができるので、セマフォをより効率的にメモリに記憶することが可能になる。

【００１７】

【発明が解決しようとする課題】コンピュータ産業においては、より高性能なハードウェアに向かって積極的な動きが続いている。しかしながら、それとは相反するように、原子的セマフォ更新を提供するよう設計されたバスロック、キャッシュ・コヒーレンシー機構および命令のエクスポートを介して原子性を提供するハードウェア・アーキテクチャも含め、幅広い多様なハードウェア・アーキテクチャ上で実行可能な、より低コストの「既製品でシュリンクラップされた（off-the-shelf shrink-wrapped）」オペレーティングシステム（およびその他のソフトウェア）に積極的に向かう傾向もある。しかし、従来技術による原子性を提供する方法は、通常、どの方法で原子性が提供されるのかをソフトウェアが「認識して」いることを当然としている。したがって、バスロックを使用してセマフォにアクセスするよう設計されたソフトウェアは、原子的セマフォ更新を提供するよう設計されたキャッシュ・コヒーレンシー機構、および命令エクスポートによって提供されるより高いセマフォのパフォーマンスを使用することができない。同様に、キャッシュ・コヒーレンシー機構を使用してセマフォにアクセスするよう設計されたソフトウェアも、原子的セマフォ更新を提供するよう設計された命令エクスポートにより提供される、より高いセマフォのパフォーマンスを使用することができない。当該技術分野において必要なのは、特定の原子的更新方法を利用するようソフトウェアを明確にコード化する必要なく、低コストの「既製品でシュリンクラップされた」ソフトウェアが、それが実行されるコンピュータシステムのハードウェアにより提供される最高のパフォーマンスの原子的更新方法にアクセスできるようにするコンピュータアーキテクチャである。

【００１８】

【課題を解決するための手段】上記の課題を解決するため、この発明は、メモリ属性フィールドをアクセスして、原子的更新処理によりアクセスされるメモリ位置が、エクスポート可能な命令をサポートしているかどうかを判断するステップと、前記原子的更新処理によりアクセスされるメモリ位置がエクスポート可能な命令をサポートしているならば、該原子的更新処理を中央ロケーションにエクスポートするステップと、前記原子的更新処理によりアクセスされるメモリ位置がエクスポート可能な命令をサポートしていないならば、キャッシュ・コ

ヒーレンシー機構を使用して原子的更新処理を実行するステップとを含む原子的更新処理を実行する方法を提供する。この発明は、バスロックを必要とするＩＡ－３２命令が、原子性を提供する優れた方法を提供するコンピュータハードウェア上で効率的に実行するような、６４ビットのアーキテクチャ・フレームワークを提供するものである。さらに、この発明は、「既製品でシュリンクラップ」のソフトウェアにコード化することのできるエクスポート可能な６４ビットのＦＥＴＣＨＡＤＤ（フェッチおよび加算）命令を定義するアーキテクチャ・フレームワークを提供し、命令をエクスポートすることにより、またはキャッシュ・コヒーレンシー機構を用いることにより、ＦＥＴＣＨＡＤＤ命令を実行する上でハードウェアが原子性を保証するプログラム可能な方法を提供する。

【００１９】ＩＡ－３２命令セットにおいては、ＬＯＣＫプレフィクスを、メモリオペランドにアクセスする形の命令に限り、それらの命令の前につけることができる。すなわち、ＡＤＤ、ＡＤＣ、ＡＮＤ、ＢＴＣ、ＢＴＲ、ＢＴＳ、ＣＭＰＸＣＨＧ、ＤＥＣ、ＩＮＣ、ＮＥＧ、ＮＯＴ、ＯＲ、ＳＢＢ、ＳＵＢ、ＸＯＲ、ＸＡＤＤ、ＸＣＨＧ命令の前につけることができる。この発明によれば、ＣＰＵは、ＩＡ－３２ロックチェック・イネーブルビット（ＬＣ）を含む省略時制御レジスタを備える。ＬＣビットが「１」に設定されており、ＩＡ－３２の原子的メモリ参照が、外部バスロック下でプロセッサの外部の「読み出し－変更－書き込み」処理を要求する（例えば、命令が、ＬＯＣＫプレフィクスを含む）とき、ＩＡ－３２インターセプト・ロックフォールトが発生し、ＩＡ－３２インターセプト・ロックフォールト・ハンドラーが呼び出される。フォールト・ハンドラーは、割り込みの原因となったＩＡ－３２命令を調べ、命令を原子的にエミュレートするために適切なコードへと分岐する。従ってこの発明は、この発明に基づいた６４ビットのアーキテクチャを持つコンピュータシステムが、ＩＡ－３２命令と２値（バイナリ）互換性を維持することができるようにし、バスをロックしないことによって、６４ビットのアーキテクチャにより提供される優れたパフォーマンスを保つことができるようにする。

【００２０】さらに、この発明は、以下のフォーマットを持つエクスポート可能なＦＥＴＣＨＡＤＤ命令を定義する。

【数１】ＦＥＴＣＨＡＤＤ　$R_1 = [R_3], INC$

【００２１】この命令は、レジスタR_3でのインデックスがついたメモリ位置を読み出し、そのメモリ位置から読み出された内容をレジスタR_1に置き、そのメモリ位置から読み出された内容にＩＮＣ値を加算し、そしてその和をそのメモリ位置に記憶しなおす。

【００２２】それぞれの仮想メモリページに関連するのは、「ライトバック方式を用いたキャッシュ可（ＷＢ）」、「キャッシュ不可（ＵＣ）」または「キャッシュ不可で、エクスポート可（ＵＣＥ）」の状態をとることができるメモリ属性である。ＦＥＴＣＨＡＤＤ命令が実行され、アクセスされたメモリ位置が、ＷＢに設定された属性を持つページにあるとき、そのメモリ位置を含むキャッシュラインの専有使用を得ることで、ＣＰＵによりＦＥＴＣＨＡＤＤ命令が原子的に実行される。しかしながら、ＦＥＴＣＨＡＤＤ命令が実行され、アクセスされたメモリ位置が、ＵＣＥに設定された属性を持つページにあるときは、ＦＥＴＣＨＡＤＤ命令を、メモリコントローラのような中央ロケーションにエクスポートすることにより、そのＦＥＴＣＨＡＤＤ命令は原子的に実行され、それによりセマフォのキャッシュラインのスラッシングを除去することができる。

【００２３】したがって、この発明は、原子性が、キャッシュ・コヒーレンシー機構により提供されるのか、またはＦＥＴＣＨＡＤＤ命令をメモリコントローラのような中央ロケーションにエクスポートすることにより提供されるのかをソフトウェアが「認識しなく」ても、ＦＥＴＣＨＡＤＤ命令によりアクセスされるセマフォで、「既製品でシュリンクラップの」ソフトウェアをコード化することのできるアーキテクチャ・フレームワークを提供する。したがって、そのようなソフトウェアは、それぞれの方法に対して個々のコードセグメントを必要とするソフトウェア無しで、コンピュータのハードウェア上で利用可能な原子的更新処理を提供する最速の方法にアクセスすることができる。

【００２４】

【発明の実施の形態】この発明は、キャッシュ・コヒーレンスを提供する優れた方法を提供するコンピュータハードウェア上で、バスロックを必要とするＩＡ－３２命令が効率的に実行する６４ビットのアーキテクチャ・フレームワークを提供する。さらに、この発明は、「既製品でシュリンクラップ」のソフトウェアにコード化することのできるエクスポート可能な６４ビットのＦＥＴＣＨＡＤＤ命令を定義するアーキテクチャと、命令をエクスポートすることにより、またはキャッシュ・コヒーレンシー機構を用いることにより、ＦＥＴＣＨＡＤＤ命令を実行する上で原子性をハードウェアが保証することのできるプログラム可能な方法を提供する。

【００２５】図１はコンピュータシステム１０の概略図であり、この発明を説明するのに使用される。コンピュータシステム１０は、ＣＰＵ１２および１４のようなＮ個のＣＰＵを備える。また、システム１０は、メモリコントローラ１６およびメインメモリ１８を備える。メモリコントローラ１６は、エクスポート可能なＦＥＴＣＨＡＤＤ命令の実行をサポートする。

【００２６】以下にＣＰＵ１２および１４について説明する前に、この発明に従って定義されるＦＥＴＣＨＡＤＤ命令について最初に述べる。この命令は、以下のフォ

ーマットを持つ。

【数2】FETCHADD R_1 = [R_3], INC

【0027】この命令は、レジスタR_3のインデックスがついたメモリ位置を読み出し、そのメモリ位置から読み出した内容をレジスタR_1に置く。さらに、この命令は、値INCを、そのメモリ位置から読み出した内容に加算し、その和をそのメモリ位置に記憶しなおす。上記のFETCHADD命令の表現は簡略化したものである。追加の命令「コンプリーターズ（completers）」は、メモリから読み出されるべきオペランドのサイズ、他の命令に対するその命令の順序づけセマンティクス（ordering semantics）、およびFETCHADD命令をCPUキャッシュにプリフェッチ（先読み）するときに使用されるプリフェッチヒントなどのようなオプションを指定する。しかしながら、この発明を理解するには上記の命令のフォーマットで充分である。

【0028】図2は、図1のCPU12のブロック図である。当然ながら、図2は、コンピュータシステム10におけるすべてのCPUを代表する。CPU12には、命令レジスタ20、命令解読実行ロジック22、フォールト・ハンドラー・ポインタ24、プログラムカウンタ26、ソフトウェアベースのIA-32インターセプト・ロックフォールト・ハンドラー28、省略時制御レジスタ（DCR）30、変換索引バッファ（TLB）36、L1およびL2キャッシュメモリ40を備える。図2は概略図であり、この発明を実現するCPUがこれよりも著しく複雑であることは当業者には明らかであろう。しかしながら、図2は、この発明の新規な側面を説明するのには充分である。

【0029】当該技術分野で知られているように、ほとんどのコンピュータシステムは、実際に存在する物理メモリより多くのメモリがあるようシミュレートする仮想メモリと呼ばれる技術を用いる。メインメモリアドレスに対する仮想アドレスのマッピングは、仮想アドレス変換として知られるプロセスである。仮想アドレスおよび物理アドレス空間は、典型的にはページと呼ばれる等サイズのメモリブロックに分割され、ページテーブルが、仮想アドレスおよび物理アドレスの間の変換を行う。それぞれのページテーブルのエントリは、典型的には物理アドレスと、ページに関する保護および状態情報とを含＊む。保護および状態情報は、典型的にはページが受けたアクセスの種類についての情報およびページ保護情報を含む。例えば、ダーティビットは、そのページのデータに対して変更が加えられていることを示す。通常、ページテーブルは大きいのでメモリに記憶される。従って、それぞれの規則的なメモリアクセスは、少なくとも2つのアクセスを実際には要求することができ、1つは変換を得るためのものであり、2つめは物理メモリ位置にアクセスするためのものである。

【0030】仮想アドレス変換をサポートするほとんどのコンピュータシステムは、変換索引バッファ（TLB）を使用する。TLBは、典型的には小容量で高速の連想メモリであり、CPU上またはCPUの近傍に通常は位置し、最近使用された仮想および物理アドレスの対を記憶する。TLBは、ページテーブルにおける変換のサブセットを含み、より高速にアクセスされることができる。処理装置は、メインメモリから情報を必要とするとき、仮想アドレスをTLBに送る。TLBは、仮想アドレスのページ番号を受け取り、物理ページ番号を返す。物理ページ番号は、メインメモリにおける所望のバイトまたはワードをアクセスするため、下位のアドレス情報と組み合わされる。その仮想アドレスの変換がTLBに無いならば、ページテーブルから抽出される。ページテーブルにその変換が無いならば、ページフォールトが生成される。

【0031】この発明によると、TLB36は、TLBエントリにより表わされるページに対して読み出しおよび書き込みを行うFETCHADD命令を、エクスポート可能であるか否かを決めるメモリアクセスビットを含む。例えば、TLBエントリ38は、メモリアクセスビットフィールド44を含む。上述したように、TLBは通常、ページテーブルに含まれる仮想ー物理マッピングのサブセットを含んでいる。従って、この発明で使用するのに適合したページテーブルも、メモリアクセスビットのコピーを含む。

【0032】表1は、メモリアクセスビットにより表すことができる仮想アドレッシングメモリ属性のコード化を示す。

【表1】

メモリ方式	ニーモニック	コード
ライトバック	WB	00
キャッシュ不可	UC	01
キャッシュ不可ーエクスポート可能	UCE	10

【0033】表1に示されるコードは、この発明をよりわかりやすく説明するため簡略化されたものであることに注意されたい。他の実施形態においては、追加の機能性をメモリアクセスビットにコード化するのが望ましいこともある。

【0034】これらのコードがどのように使用されるの※かを説明するため、CPU12、またはCPU12が作動するコンピュータシステムのいずれも、エクスポート可能のFETCHADD命令をサポートしていない場合を想定する。そのようなコンピュータシステムにおいては、原子性は、キャッシュ・コヒーレンシーのアルゴリズムによって提供される。セマフォを含むメモリ位置に

ついて仮想－物理マッピングが確立されると、ページテーブルにおけるメモリアクセスビット、およびセマフォが記憶されたページに対応するＴＬＢエントリが、ＷＢのメモリ方式に指定され、従ってメモリアクセスビットが「００」に設定される。

【００３５】従って、ＦＥＴＣＨＡＤＤ命令が命令レジスタ２０にロードされると、命令解読実行ロジック２２は、ＦＥＴＣＨＡＤＤ命令により指定されたメモリ位置に対応するＴＬＢエントリに記憶されたメモリアクセスビットを調べる。メモリアクセスビットは、ライトバック方式を用いてキャッシュされたメモリページにセマフォが記憶されることを示すので、セマフォを含むキャッシュラインがＬ１およびＬ２キャッシュ４０にロードされ、専有として保持される。その後、命令解読実行ロジック２２は、セマフォをＬ１キャッシュから抽出し、そのセマフォを、ＦＥＴＣＨＡＤＤ命令中に指定されたレジスタファイル４２のレジスタにロードし、セマフォをインクリメントし、インクリメントされたセマフォをＬ１キャッシュに記憶し直す。セマフォが他のＣＰＵにより要求されるとき、ＣＰＵ１２は、セマフォを含むキャッシュラインの専有使用を放棄し、他のＣＰＵがそのキャッシュラインの専有使用を獲得する。これにより、ある量のキャッシュラインのスラッシングとなるけれども、パフォーマンスは、ローエンドからミドルレンジのコンピュータシステムにおいては充分過ぎると言える。

【００３６】次に、ＣＰＵ１２、およびＣＰＵ１２が作動するコンピュータシステムの両方が、エクスポート可能なＦＥＴＣＨＡＤＤ命令をサポートする場合を想定する。そのようなコンピュータシステムにおいては、ＦＥＴＣＨＡＤＤ命令の原子性は、ＦＥＴＣＨＡＤＤ命令をメモリコントローラ（または、その他の中央ロケーション）にエクスポートすることによるか、またはキャッシュ・コヒーレンシー機構によるかのいずれかによって提供することができる。仮想－物理マッピングが、セマフォを含むメモリ位置について確立されるとき、ページテーブルのメモリアクセスビット、およびセマフォが記憶されているページに対応するＴＬＢエントリが、ＵＣＥメモリ方式に指定され、従ってメモリアクセスビットは「１０」に設定される。

【００３７】従って、ＦＥＴＣＨＡＤＤ命令が命令レジスタ２０にロードされるとき、命令解読実行ロジック２２は、ＦＥＴＣＨＡＤＤ命令により指定されたメモリ位置に対応するＴＬＢエントリに記憶されたメモリアクセスビットを調べる。メモリアクセスビットは、セマフォが、キャッシュ不可でエクスポート可能なメモリページに記憶されていることを示す。従って、命令解読実行ロジック２２は、ＦＥＴＣＨＡＤＤ命令を図１のメモリコントローラ１６にエクスポートする。コントローラ１６は、図１のメインメモリ１８からセマフォを読み出し、そのセマフォを命令解読実行ロジック２２に提供し、命令解読実行ロジック２２は、ＦＥＴＣＨＡＤＤ命令中に指定されたレジスタファイル４２のレジスタにセマフォを記憶する。メモリコントローラ１６は、セマフォをインクリメントし、結果をメインメモリ１８に記憶しなおす。セマフォがキャッシュラインに専有として保持されることが無いので、他のＣＰＵは、セマフォを含むキャッシュラインの専有使用を得る必要なく、即座にセマフォにアクセスすることができる。従って、キャッシュラインのスラッシングが除去される。メモリコントローラ１６が、ＦＥＴＣＨＡＤＤ命令によりアクセスされるセマフォのキャッシュを保持するのが好ましく、これにより、メモリコントローラ１６は、メインメモリ１８にアクセスする必要がなくなってより速い応答が可能となる点に注意されたい。

【００３８】要約すると、この発明は、原子性がキャッシュ・コヒーレンシー機構により提供されるのか、またはＦＥＴＣＨＡＤＤ命令をメモリコントローラ１６のような中央ロケーションへとエクスポートすることにより提供されるのかを「既製品でシュリンクラップの」ソフトウェアが「知らない」場合でも、該ソフトウェアを、ＦＥＴＣＨＡＤＤ命令によりアクセスされるセマフォでコード化することのできるアーキテクチャ・フレームワークを提供する。従って、このようなソフトウェアは、それぞれの方法について個々のコードセグメントを必要とするソフトウェアなしで、コンピュータハードウェア上で利用可能な原子性更新処理を提供するもっとも高速な方法をアクセスすることができる。

【００３９】この発明により提供される他の利点は、ＦＥＴＣＡＤＤ命令のエクスポートをサポートするのに、非常に狭い範囲のメモリ位置を選択的にイネーブル（使用可能）にできるということである。従って、オペレーティングシステムは、メモリの小さな部分を、キャッシュ不可でエクスポート可能なよう構成することができ、アプリケーションプログラムが、オペレーティングシステムからセマフォ用のメモリ空間を要求するとき、オペレーティングシステムは、そのような空間を、キャッシュ不可でエクスポート可能なよう構成された領域に割り振ることができる。また、この発明は、オペレーティングシステムが、Ｉ／Ｏ装置にマッピングされたメモリ位置のような、エクスポート可能なＦＥＴＣＨＡＤＤ命令をサポートしないメモリ範囲に対するＦＥＴＣＨＡＤＤ命令のエクスポートを防ぐことができるようにする。

【００４０】この発明は、キャッシュ・コヒーレンシー機構またはＦＥＴＣＨＡＤＤ命令のエクスポートのいずれかを選択することにより原子性を提供する、６４ビットのＦＥＴＣＨＡＤＤ命令およびそれをサポートするアーキテクチャ・フレームワークを提供するけれども、この発明は、バスロックのプレフィクスを介して原子性を提供するＩＡ－３２命令をもサポートする。ＩＡ－３２命令セットの詳細は、「Intel 命令セットリファレン

ス」に見つけることができ、ここで参照により取り入れる。

【0041】IA－32の命令セットにおいて、LOCKプレフィクスは、メモリオペランドにアクセスする形の命令に限り、それらの命令の前に付けることが出来る。すなわち、ADD、ADC、AND、BTC、BTR、BTS、CMPXCHG、DEC、INC、NEG、NOT、OR、SBB、SUB、XOR、XADD、XCHG命令につけることができる。

【0042】図2を参照すると、省略時制御レジスタ（DCR）30は、IA－32ロックチェック・イネーブルビット（LC）32を含む。LCビット32が「1」に設定され、IA－32の原子的メモリ参照が、外部バスロック下でプロセッサの外部の「読み出し－変更－書き込み」処理を要求する（例えば、命令がLOCKプレフィクスを含む）とき、IA－32インターセプト・ロックフォールトが起こる。LCビット32が「0」にクリアされ、IA－32原子的メモリ参照が、外部バスロック下でプロセッサの外部の「読み出し－変更－書き込み」処理を要求とするとき、プロセッサは、IA－32インターセプト・ロックフォールトを生成するか（バスロックが、コンピュータシステムのハードウェアによりサポートされていない場合）、または外部バスロックでトランザクションを実行することができる。IA－32アーキテクチャにおいては、ライトバックのキャッシュ方式を使用してキャッシュされないメモリに対して行われる原子的メモリアクセスには、外部バスロックが必要となる点に注意されたい。言い換えると、メモリがキャッシュ不可またはライトスルー方式でキャッシュされる場合には、IA－32アーキテクチャには外部バスロックが必要となる。

【0043】複数の相互接続トポロジーで接続された複数のプロセッサを持つコンピュータシステムのように、バスをロックすることができないコンピュータシステムにおいては、当然ながら、LOCKプレフィクスが前についたIA－32命令は、IA－32インターセプト・ロックフォールトにより取り扱われなければならない。そのようなシステムでは、LCビット32の値は問題とならない。

【0044】しかしながら、バスをロックすることができるコンピュータシステムを考えてみる。バスをロックすることが可能であるけれども、継続的にそうすることによりパフォーマンスが著しく低下することがある。従って、バスをロックしない命令を使ってLOCKプレフィクスのついたIA－32命令をエミュレートするのが望ましい。エミュレーションは、ハードウェアによって直接実行する程速くはないが、LOCKプレフィクスのついたIA－32命令がエミュレートされる間に他のCPUおよび装置がアクセスを続けることができるので、システム全体のパフォーマンスを上げることができる。このようなシステムにおいては、LOCKプレフィクスのついたIA－32命令が、バスロックを使用してハードウェアにより実行されるのか、またはバスロックを使用せずにソフトウェアでエミュレートされるのかを、LCビット32の値が決定する。LOCKプレフィクスのついたIA－32命令をエミュレートするとき、当然ながら、エミュレーションコードは、キャッシュ・コヒーレンシー機構および／またはエクスポート可能なFETCHADD命令を使用して原子性を確保することができる。

【0045】図2に戻ると、ADD、ADC、AND、BTC、BTR、BTS、CMPXCHG、DEC、INC、NEG、NOT、OR、SBB、SUB、XOR、XADDおよびXCHGの命令セットからのIA－32命令が、LOCKプレフィクスが前につけられており、メモリ位置をアクセスし、命令レジスタ20にロードされると想定する。さらに、DCR30のLCビット32が「1」に設定されていると想定する。このような命令に応答して、命令解読実行ロジック22は、IA－32インターセプト・ロックフォールトを生成する。ロジック22は、フォールト・ハンドラー・ポインタ24におけるIA－32インターセプト・ロックフォールトエントリ34に記憶されたアドレスを、プログラムカウンタ26にロードする。これにより、フォールト・ハンドラー28の最初の命令が命令レジスタ20にロードされ、ソフトウェアベースのIA－32インターセプト・ロックフォールト・ハンドラー28の最初の命令が実行される。フォールト・ハンドラー28は、割り込みを引き起こしたIA－32命令を調べ、原子的にその命令をエミュレートするため適切なコードへと分岐する。IA－32命令がエミュレートされ、適切な値がレジスタファイル42、L1およびL2キャッシュ40および図1のメインメモリ18に記憶された後、フォールト・ハンドラー28は終了し、フォールトを引き起こした命令のすぐ後の命令を続けて実行する。

【0046】フォールト・ハンドラー28が、上述したようなエクスポート可能なFETCHADD命令を含むことができる点に注意されたい。例えば、宛先メモリ位置を「1」だけインクリメントするようコード化されたIA－32XADD命令を、「1」に設定されたインクリメント値（INC）を持つFETCHADD命令によりエミュレートすることができる。FETCHADD命令によりエミュレートすることのできない他の命令については、変更されるべきメモリ位置を含むキャッシュラインの専有使用を得る命令によりエミュレートすることができる。

【0047】要約すると、DCR30のLCビット32およびソフトウェアベースのIA－32インターセプト・ロックフォールト・ハンドラー28により、CPU12は、バスをロックすることなくLOCKプレフィクス

のついたＩＡ－３２命令を原子的に実行することができるようになる。従って、この発明により、６４ビットのアーキテクチャを持つコンピュータシステムは、６４ビットのアーキテクチャにより提供される優れたパフォーマンスを維持しつつ、ＩＡ－３２命令との２値互換性を維持することができるようになる。

【００４８】この発明を、好ましい実施形態を参照しつつ説明してきたけれども、当該技術分野の当業者には、この発明の精神および範囲から離れること無く、形式および詳細において変更を加えることができるということが明らかであろう。

【発明の効果】特定の原子的更新方法を利用するようソフトウェアを明確にコード化することなく、ソフトウェアは、ハードウェアにより提供される最高のパフォーマンスの原子的更新方法をアクセスできるようになる。

【図面の簡単な説明】

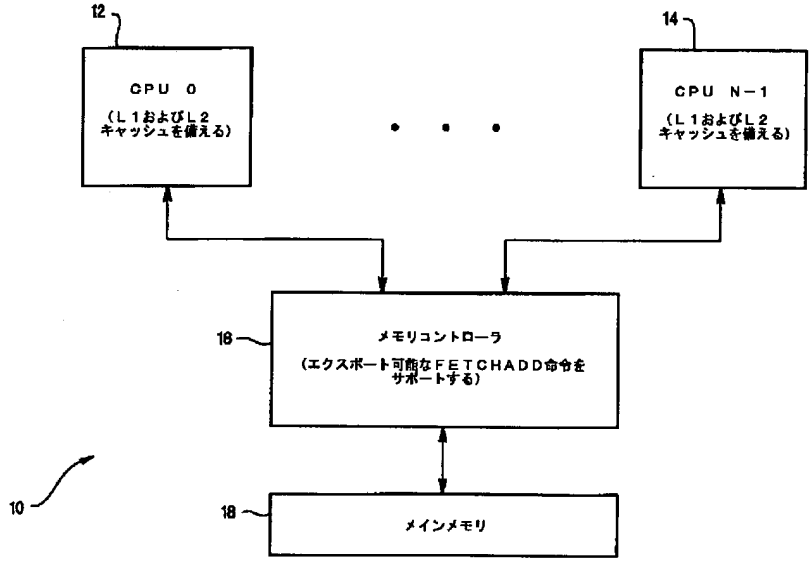
【図１】Ｎ個のＣＰＵ、メモリコントローラおよびメインメモリを備えるコンピュータシステムの概略図。

【図２】この発明による、図１のＣＰＵのうちの１つのブロック図。

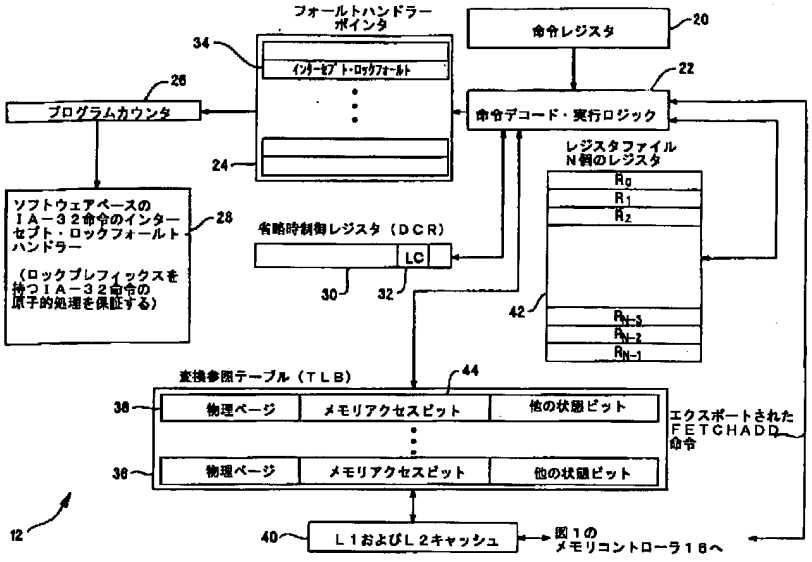
【符号の説明】

１２、１４　ＣＰＵ
１６　メモリコントローラ
１８　メインメモリ
３６　ＴＬＢ
４４　メモリ属性フィールド

【図１】



【図2】



フロントページの続き

(72)発明者　ミラード・ミッタル
アメリカ合衆国94080カリフォルニア州サウス・サン・フランシスコ、ヒルサイド・ブルバード　1149

(72)発明者　マーティン・ジェイ・ウイッテカー
アメリカ合衆国95014カリフォルニア州クパーチノ、ストニーデイル・ドライブ　10241

(72)発明者　ガリー・エヌ・ハモンド
アメリカ合衆国95008カリフォルニア州キャンベル、サニーブルック・ドライブ　519

(72)発明者　ジェローム・シー・ハック
アメリカ合衆国94303カリフォルニア州パロ・アルト、タリスマン・ドライブ　851